The Expeditious Cipher (X-cipher): A High Throughput Lightweight Cipher

Magdy Saeb

Arab Academy for Science, Technology & Maritime Transport

Computer Engineering Department

Alexandria, EGYPT

mail@magdysaeb.net

Abstract: Pervasive computing utilizes devices with cost constraints and limited resources in terms of memory, processing speed, and battery supply. Furthermore, we are witnessing the phenomena of halving of the price for constant computing power in less than two years. RFID processes sensitive health-monitoring or biometric data. Accordingly there is a demand for specially-designed cryptographic primitives that are to be efficiently implemented. For these implementations and ciphers that are particularly suited for this purpose, we use the term "lightweight cipher".

The Expeditious Cipher (X-cipher) is a high throughput light-weight cipher with relatively simple design and hardware implementation. In this article, we discuss the cipher design rationale, its software and hardware implementations. The cipher security is based on the well-studied SHA-512 and Whirlpool 512 hash functions that are used to generate the cipher sub-keys. We appreciably enhance the cipher security by cascading the two hashes in a pseudo random manner that is user-key-dependent. The cipher employs a variable-size user key and we show that the encryption rate for its associated hardware implementation reaches 512 bits per cycle. The block size can be varied using other types of hash functions.

1. Introduction

Pervasive computing applies devices with cost constraints and limited resources in terms of memory, processing speed, and battery supply. Furthermore, we are witnessing the phenomena of halving of the price for constant computing power in less than two years. RFID processes sensitive health-monitoring or biometric data. Accordingly there is a demand for specially-designed cryptographic primitives that are to be efficiently implemented. For these implementations and ciphers that are particularly suited for this purpose, we use the term "lightweight cipher". The notion of a light weight cipher design was discussed in quite a few articles [1, 2, 3, 4]. In designing lightweight ciphers one has to think of some trade-off between security, cost, and performance. In a large number of cases, it is unmanageable to optimize all three design objectives. For example, a secure and high-performance hardware implementation can be achieved by a pipelined architecture that is side-channel attack resistant. This, in turn, results in a large implementation area and thus high cost. On the other hand, it is possible to design a secure and low-cost hardware implementation, nevertheless, of bounded performance.

The Expeditious Cipher (X-cipher) is a high throughput light-weight cipher with a relatively simple design and efficient hardware implementation. In this article, we discuss the rationale of the cipher design, its software and hardware implementations. The cipher security, we show, is based on the well-studied SHA-512 and Whirlpool-512 hash functions, which are used to generate the cipher sub-keys.

Received 8/10 Reviewed 8/23

Moreover, we enhance the cipher security by cascading the two hashes in a pseudo random manner that is user- key-dependent. We show that the cipher employs a variable-size user key and that the encryption rate for its hardware implementation, with the parallelization encryption paths, can reach 512 bits per cycle. The block size can be varied using other types of hash functions. In the following sections we discuss the design rationale, cipher structure, the algorithm, the sub-key generation, the cipher security and performance. Finally, we provide a summary and our conclusions.

2. Design Rationale

A cipher can be simplified into a switching circuit, as shown in Figure 1, which is controlled by a pseudo random number generator PRG that inverts/does not invert a plaintext bit. If the PRG is properly designed, the resulting cipher bits will be inverted by a ratio of approximately 50% increasing the entropy to almost to a maximum value. For practical ciphers, usually this ratio is slightly less than 50% around 47-49%.

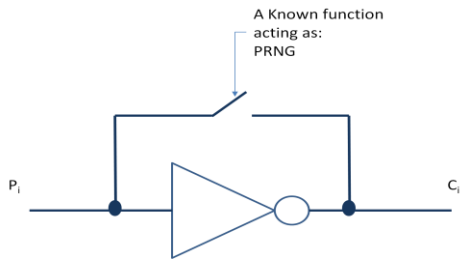
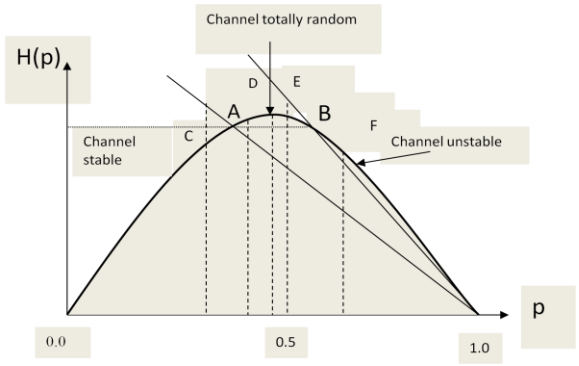
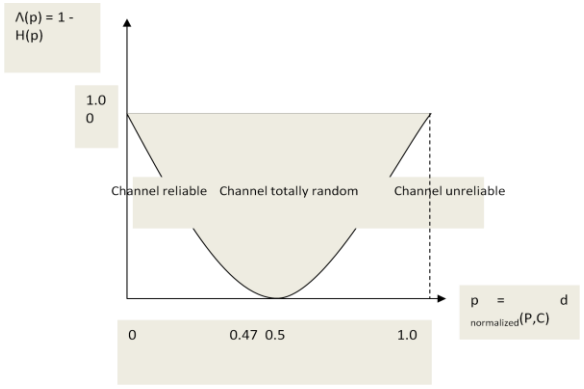


Figure 1: *Switch*

This ratio of the number of bits changed can be easily calculated by measuring the Hamming distance between the plaintext bits and the cipher bits. The graph, shown in Figure 2, clearly demonstrates this notion. Moreover, this number of changed bits determines the operating point of the cipher on the entropy versus number of changed bits plot shown in Figure 2. In this plot H (p) is the entropy and p is the ratio of bits changed between the plaintext and the cipher text. For maximum entropy this ratio is clearly shown to be 0.5. However, most ciphers produce a ratio of less than 0.5 rather than greater than 0.5 since the channel in this case will be unstable, as shown in Figure 2. For example points A and B on the entropy plot have the same entropy; however point A is in the stable region of the plot while point B is in the unstable region. Therefore, all ciphers and hash functions operate in this region. When the entropy is maximized, at the point corresponding to p = 0.5, this is considered a critically stable point and the channel is totally random with its capacity equal to zero as shown in Figure 2.b.



2.a: *The Entropy H(p) versus number of changed bits plot indicating the stable and the unstable channel regions of operation,*



2. b: *The channel capacity given by* $\Lambda(p) = 1 - H(p)$

Figure 2: *The Entropy (2.a), and the Channel Capacity (2.b)*

The PRG design determines the security and the strict avalanche criteria (SAC) requirements of any cipher or hash function. The hardware implementation of such a switch and the controlling parameter of the PRG is shown in Figure 3. This parameter or the external source of entropy is the round key. The round key is extracted from the user key using any one of the following well-known techniques:

- Using a given function
- Using the cipher itself
- Using a hash function

.

In this design, we use standard hash functions with well-established security properties to generate the

sub-keys. These sub-keys are then utilized to turn on or turn off the bit inversion switching path. The realization or implementation of such an encryption switch using tri-state buffers is shown in Figure 3.

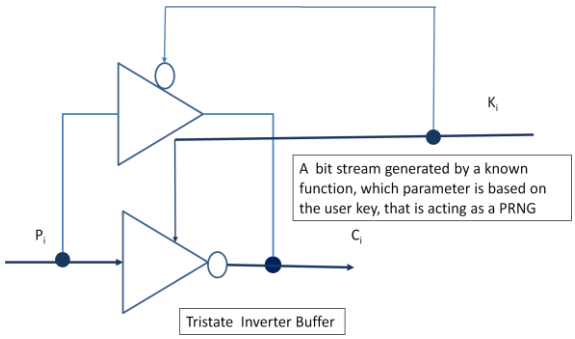


Figure 3: *The Encryption Switch realization*

Checking the truth table of such a switching device, shown in Table 1, one can immediately recognize that this switch is logically equivalent to an XOR logic gate. This type of switching provides no bias in the output cipher bits to either the zeros or the ones. That is the main reason that an XOR logic function is an essential part of any presently known cipher.

Table 1: *The truth table of the switching device*

P_i	K_i	C_i
0	0	0
0	1	1
1	0	1
1	1	0

In order to enhance the cipher security, one selects a certain number of bits from the user key, or even better, from its hashed value to determine the sequence of hashing the user key to provide the sub-keys. We call this series of bits "the bit pattern". For an example if the bit pattern is, say, 101101, then the first sub-key will be given by:

$$K_1 = h_1 (h_0 (h_1 (h_1 (h_0 (h_1 (K_h))))))),$$

Where h_0 and h_1 are two hash functions, of the same output number of bits and K_h is the hashed user's key. This thought of replacing one hash function by two hashes, pseudo randomly cascaded, appreciably enhances the cipher algorithm. Moreover, it agrees with the notion of key-dependent ciphers [5, 6]. The idea can be also generalized for more than two cascaded hash functions, however, this may be considered to be unnecessary for most data security applications. Other sub-keys can be also generated by adding a large, agreed upon, number and then go through the same procedure to generate the next sub-key.

3. The Cipher Structure

The cipher is built using three building blocks:

- The hash block that hashes the user key into a 512 bit-output.
- The PRG built of two hash functions, which generates the round keys. The two hash functions utilized in this cipher are the SHA-512 and the Whirlpool 512 hash functions.
- The set of registers or temporary storage locations for the generated sub-keys, the plain text block and resulting ciphertext block.
- The set of 512 XOR logic gates which input are connected to the plaintext register and the key bit register. Their output is connected to the cipher block register.

This cipher structure is shown in Figure 4. The curved arrow represents the next round of the cipher. In this cipher, based on our experience gained from cipher testing, we recommend that the cipher uses eight rounds. From this structure, one can readily notice that a block of 512 bits is executed in parallel. Thus the encryption rate is slightly larger than 512 bits per cycle taking into consideration the setup and hold times of the registers employed.

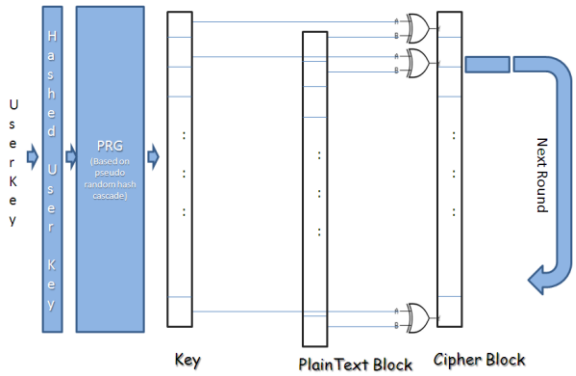


Figure 4**:** *The Cipher Structure*

4. The Algorithm

The algorithm can be formally described as shown in the next few lines.

Algorithm: EXPEDITIOUSCIPHER
INPUT: Plain text message P, User Key K, Block Size B = 512 bits, M is an agreed-upon large integer number, SHA-512, Whirlpool-512 hash functions..
OUTPUT: Cipher Text C
Algorithm body:
Begin
Begin key schedule
1. Read user key;
2. Encrypt user key by calling hash function SHA-512.;
3. Read the values of the bit pattern from an agreed-upon field of the resulting hashed user key;
4. Generate a sub-key by calling the pseudo randomly generated ordered cascade of the two hash functions;
5. Store the generated value of the subkey;
6. Add a large number M to sub-key obtained;
7. Repeat steps 4 to 6 to generate the next sub-key:
End key schedule;
Begin Encryption
8. Read a block size equal to 512 bits of the message P into the message cache;
9. Use the generated 512-bit sub-key to bit-wise encrypt the plain text bits using an XOR logic operation between the key bits and the plaintext block bits;
10. If message cache is not empty, Go to step 8;
11. Else if message cache is empty:
If message not finished
10.1 Load next block into message cache;
10.2 Go to 8;
Else if message is finished then halt;
End Encryption;
End Algorithm.

5. Sub-Key Generation

As discussed in sections 2 and 3, the sub-keys are generated using a pseudo randomly generated cascade of two hash functions. The bit pattern used is derived from the hashed user key. If this pattern is, say, $a_5a_4a_3a_2a_1a_0$ then the sub-keys are generated using the following relation:

$$K_{i+1} = h_{a5}(h_{a4}(h_{a3}(h_{a2}(h_{a1}(h_{a0}(K_i + M))))))),$$

Where K_i is the sub-key for i-th round, M is an agreed-upon large integer number and $a_i \varepsilon \{1,0\}$ that depends on the selected pseudo random bit-pattern. This relation can be generalized for more than two hash functions and the bit pattern can be altered for each sub-key when it is taken from the previous key. However, for most present-day data security requirements this generalization is hardly required. The time required to perform this operation can be estimated by adding the execution times of the two selected hash functions based on the previously discussed bit pattern. However, the expected time can be computed from the following relation:

Expected execution time for the sub-key generation is given by:

$$T_{\text{sub-key}} = 3.\ N.\ (h_0 \text{ execution time} + h_1 \text{ execution time})$$

We take N as the number of the cipher rounds. This sub-key generation time, while it is relatively large when compared to the encryption time, yet it is performed only once independent of other cipher parameters. The specifications of the hash function used can be found in [7] and the analysis of cascaded ciphers in [8].

6. Security & Performance

Based on testing results performed on SHA-512, it was found [9] that the hash time varies depending on the hashed file length. For example for a file length of 1024 bits, the hash time was found to 0.000115307921091609 second and for a 1 Mbit file, the hash time is 0.024766415843354400. Based on these tests, one can estimate the average hashing time to be approximately equal to 70 nano sec/bit hashed. Then for a file length of 1 Mbits, the execution time, assuming equal times for SHA-512 and Whirlpool-512 is equal to 24 * 1E6 *70 E-9 *2= 3.36 s which is an acceptable value for a one-time key generation execution time. The expected encryption time for a 512-bit block is slightly greater than one cycle taking

into consideration the setup and hold times required for the registers storing the plaintext, key and cipher. This time is about 0.3344481605 pico sec for 2.99GHz machine. However, for a 100 MHz FPGA implementations this number will increase to 10 nano second, which is acceptable for most applications. Another set of test results [10] show that the cycles per byte for SHA-512 is 17.7 and for Whirlpool is 33.5. Accordingly for a file size of a 1Mbyte file, the hash time is approximately 51.2*24 M cycles. The cycle time for 3 GHz machine, the hash time is approximately 0.4096 second. However, for a 100 MHz FPGA implementation this time is estimated to be about 12.288 second. That is probably practical for some applications. However, increasing the FPGA system frequency to 300 MHz will reduce this time to 4.096 second. The encryption time in this case is slightly more than 3.33 nano second for a 512-bit block.

7. Summary & Conclusions

The Expeditious Cipher (X-Cipher) and the rationale behind its design have been presented. It is now an accepted thought that design and cryptanalysis go in collaboration [11, 12]. It is inconceivable to do one without the other. It is only in the analysis that the strength of an algorithm can be demonstrated. Therefore, we have proposed the notion of separating the encryption process and the key scheduling process. Using standard hash functions for key scheduling process allows for the burden of proof to tend to lie with their designers. Accordingly, I reason the following:

- Using standard hash functions for sub-key generation, such as SHA-512, and whirlpool-512, allows the cipher security to be totally dependent on these hash functions. Therefore, the cipher security proof is totally dependent on the security proofs of these two hash functions.

- Separating the sub-key generation, or what is usually referred to as the key scheduling process, from the encryption process allows changing the cipher design, block and key sizes simply by changing the hash function employed.

- Selecting a key-dependent pseudo randomly ordered cascade of two hash functions enhances the cipher security. In addition, the pattern utilized to generate this order can be taken from previous sub-keys and consequently heightens security tremendously. However, this is deemed as unnecessary for present-day data security requirements.

- The cipher is marked by speed and efficiency, since the sub-key generation is performed only once while the encryption process speed is almost one block per cycle. In our case this is 512 bits per cycle.

- Depending on the available storage of the device utilized, one can increase or decrease the number of sub-keys generated and accordingly the number of rounds. Thus, providing security and hardware cost trade-offs as mentioned in the introduction of this article.

- The cipher simple design provides a platform for efficient hardware implementation where the cipher throughput is high (512 bits per cycle) and the implementation area of the encryption process is minimized. However, a relatively large area will be consumed in the hash function implementation.

- The encryptor and decryptor code or circuitry are identical, therefore we get high code density and relatively small implementation area.

The cipher is marked by speed and efficiency and adaptability to hardware implementation. Its security is totally dependent on the hash functions used. I hypothesize that the most efficient way to break "The Expeditious Cipher" is through exhaustive search of the key space. I encourage all cryptanalytic attacks to enrich our perception, learning and reasoning.

References

1. A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. B. Robshaw, Y. Seurin and C. Vikkelsoe
"PRESENT: An Ultra-Lightweight Block Cipher," Cryptographic Hardware and Embbeded Systems-CHEZ 2007, Lecture Notes in Computer Science, Volume 4727/2007, 450-466, DOI: 10.1007/978-3-540-74735-2_31, 2007. 2.

2. Axel Poschmann , Gregor Le , Kai Schramm , Christof Paar, "A Family of Light-Weight Block Ciphers Based on DES Suited For Rfid applications," Proceedings of FSE, LNCS, 2007.

3. Thomas Eisenbarth, Sandeep Kumar, Christof Paar and Axel Poschmann, Leif Uhsadel, "A Survey of Lightweight-Cryptography Implementations," IEEE Design & Test of Computers, 2007.

4. Wenling Wu and Lei Zhang, "LBlock: A Lightweight Block Cipher," ACNS, LNCS 6715, pp. 327-344, 2011.

5. Magdy Saeb, "The Stone Cipher-192 (SC-192): A Metamorphic Cipher," (IJCNS) International Journal of Computer and Network Security, *Vol. 1, No. 2, November 2009.*

6. Magdy Saeb, "Hardware Implementation of the Chameleon Polymorphic Cipher-192," IJCSNS International Journal of Computer Science and Network Security, VOL.9 No.11, November 2009.

7. FIPS PUB 180-3, FEDERAL INFORMATION PROCESSING STANDARDS PUBLICATION, Secure Hash Standard (SHS), Information Technology Laboratory
National Institute of Standards and Technology Gaithersburg, MD 20899-8900, October 2008.

8. Peter Ga˘zi1, and Ueli Maurer, "Cascade Encryption Revisited, "ASIACRYPT 2009, LNCS 5912, pp. 37–51, 2009.

9. http://blog.dicelocksecurity.com/visual-studio-c-2008/established-sha-512-secure-hash-algorithm-execution-time-relation-bit-stream-length/ Accessed, October, 8, 2011.

10. http://www.cryptopp.com/benchmarks.html Accessed, October, 8, 2011.

11. Krystian Matusiewicz, Analysis of Modern Dedicated Cryptographic Hash Functions, Ph.D. Dissertation, Department of Computing, Division of Information and Communication Sciences, Macquarie University, August 2007.

12. Julia, Borghoff, Cryptanalysis of Lightweight Ciphers, Ph.D. Dissertation, Technical University of Denmark, DTU Mathematics, Department of Mathematics, December, 2010.

Magdy Saeb received the BSEE., School of Engineering, Cairo University, in 1974; the MSEE. and Ph.D. in Electrical & Computer Engineering, University of California, Irvine, in 1981 and 1985, respectively. He was with Kaiser Aerospace and Electronics, Irvine California, and The Atomic Energy Establishment, Anshas, Egypt. Currently, he Magdy Saeb is a professor in the Department of Computer Engineering, Arab Academy for Science, Technology & Maritime Transport, Alexandria, Egypt, (was on leave) to Malaysian Institute of Microelectronic Systems (MIMOS), Kuala Lumpur, Malaysia. His current research interests include Cryptography, FPGA Implementations of Cryptography and Steganography Data Security Techniques, Mobile Agent Security, Bioinformatics. . www.magdysaeb.net